D Y PATIL INTERNATIONAL UNIVERSITY
School of Continuing Education

B.TECH MECHANICAL ENGINEERING

2D and 3D Transformations in CAD

"From mathematical derivations to advanced matrix operations"

SESSION

January - May 2026

6th Semester

PREPARED BY

Dinesh Kumar

Assistant Professor

Table of Contents

Comprehensive guide to CAD Geometric Transformations

Why Transformations Matter in CAD

Core geometric operations powering modern design and visualization

Reposition & Orient Assemblies

Transformations allow individual parts to be moved and rotated into their correct positions within a complex assembly without redefining geometry.

Multi-view Visualization

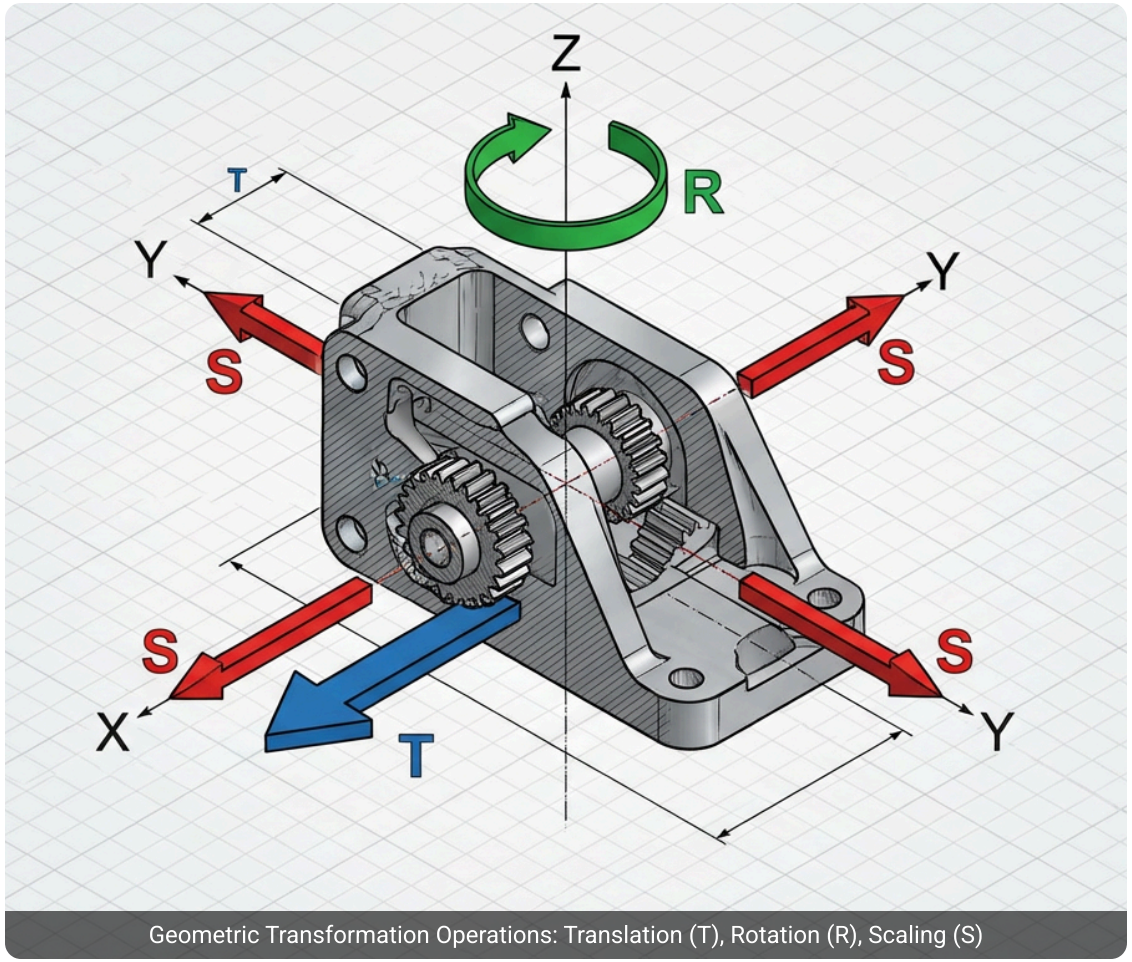
Enables viewing models from any angle (top, front, isometric) by transforming the object relative to the camera or viewing plane.

Geometric Reusability

Define a component once in its local coordinate system and instantiate it multiple times at different locations and scales throughout the design.

Numerical Robustness

Matrix-based operations ensure precision and allow complex sequences of movements to be computed efficiently as a single composite operation.



Geometric Transformation Operations: Translation (T), Rotation (R), Scaling (S)

Coordinate Systems: MCS, WCS, SCS

Fundamental reference frames for geometry definition and display

MCS (Model)

Model Coordinate System: Local reference frame attached to an individual object. Geometry is defined here (e.g., origin at center of mass).

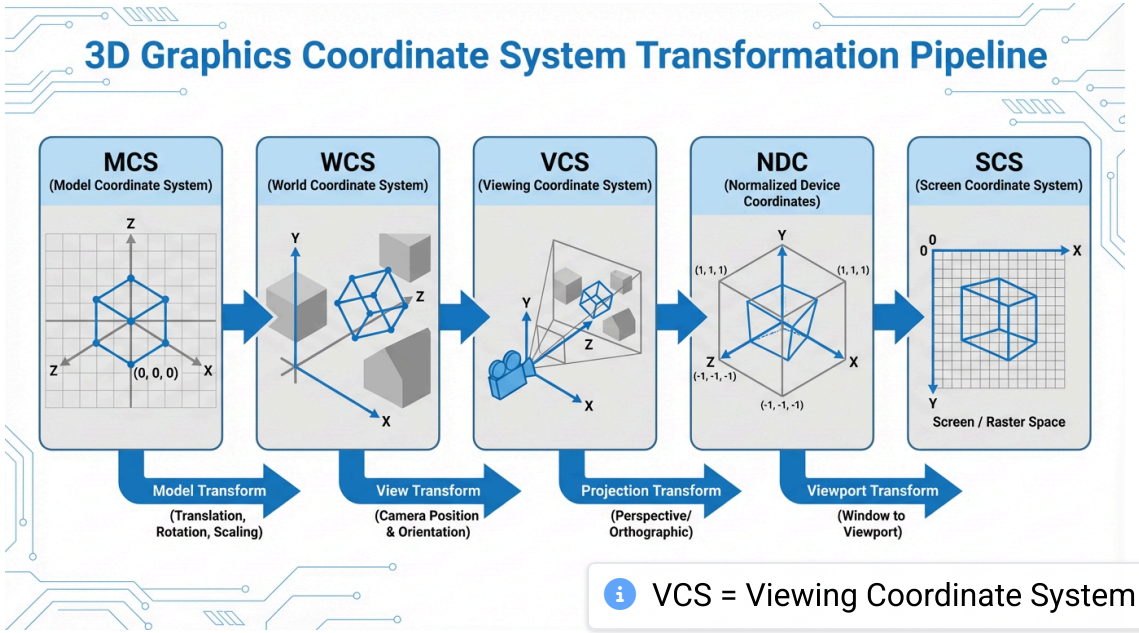
WCS (World)

World Coordinate System: Global reference frame for the entire scene. Relates multiple MCS instances to each other in assembly.

SCS (Screen)

Screen Coordinate System: 2D device-dependent system (pixels). Origin usually at top-left or bottom-left of the display.

Coordinate Transformation Pipeline



VCS = Viewing Coordinate System | NDC = Normalized Device Coordinates

Homogeneous Coordinates

Concept, Derivation, and Mathematical Benefits in CAD

Dimensional Augmentation

To represent translation as a matrix multiplication (linear operation), we add an extra dimension 'w'.

`2D: (x, y) → (x, y, 1)` `3D: (x, y, z) → (x, y, z, 1)`

Unified Transformations

Enables treating Translation, Rotation, and Scaling uniformly as matrix multiplications. This unification is crucial for creating composite transformations (concatenation).

Points vs. Vectors

Helps distinguish position from direction:

Points have w=1 (affected by translation).

Vectors have w=0 (direction only, translation invariant).

MATHEMATICAL CONCEPT

$P(x, y)$ → $P_h(xw, yw, w)$

Cartesian → Homogeneous

NUMERICAL EXAMPLE (W=2)

Input: Point (3, 4)

Homo: (3×2, 4×2, 2) = **(6, 8, 2)**

Result: x = 6/2 = 3, y = 8/2 = 4

TRANSLATION MATRIX

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Matrix multiplication enabled

2D Translation: Theory and Matrix Form

Moving objects linearly across the Cartesian plane

Theory

A translation moves an object to a different position on the screen. It shifts a point from coordinate position `P(x, y)` to a new position `P'(x', y')`.

Algebraic Equations:

Adding translation distances t_x and t_y to the original coordinates:

$$x' = x + t_x$$

$$y' = y + t_y$$

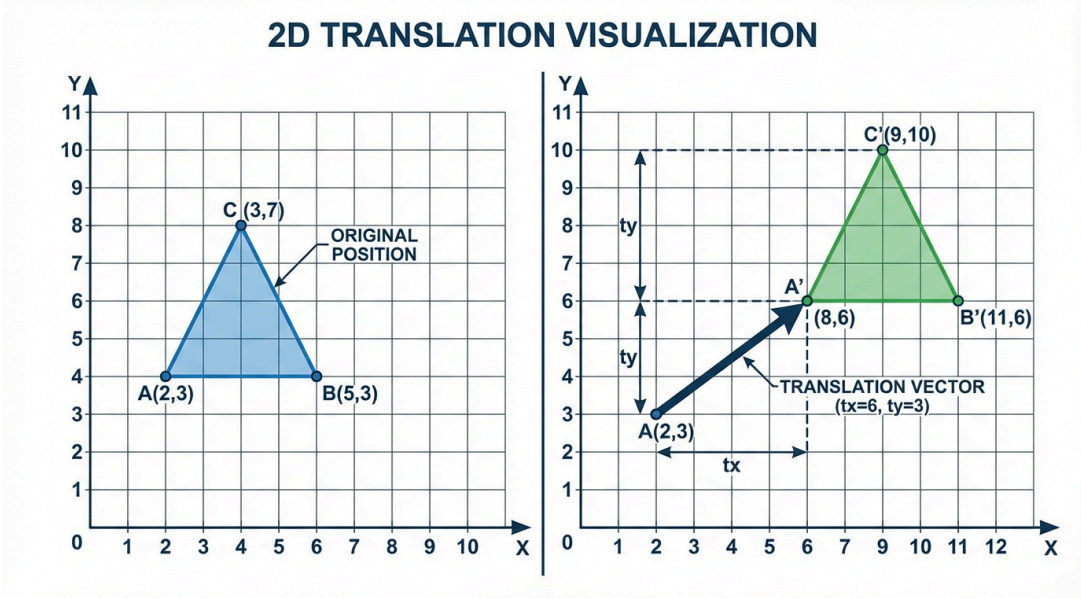
The pair (t_x, t_y) is called the translation vector or shift vector.

Matrix Representation

Using Homogeneous Coordinates to represent translation as a matrix multiplication:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Note: In compact form: $P' = T(t_x, t_y) \cdot P$. This 3x3 matrix form allows translation to be combined with rotation and scaling via matrix multiplication.



2D Translation: Numerical Examples

Applying translation equations and matrices to solve geometric problems

BASIC LEVEL

Triangle Translation

Problem: Translate triangle ABC with vertices A(2,2), B(10,2), and C(5,5) by translation vector T = (5, 6).

1. Identify translation factors: `tx = 5, ty = 6`
2. Apply equations: `x' = x + tx, y' = y + ty`
3. Calculate new coordinates:

Vertex	Operation	Result (x', y')
A(2, 2)	(2+5, 2+6)	**A'(7, 8)**
B(10, 2)	(10+5, 2+6)	**B'(15, 8)**
C(5, 5)	(5+5, 5+6)	**C'(10, 11)**

ADVANCED LEVEL

Composite Translation

Problem: A polyline is translated by T1(-3, 4) and then by T2(8, -2). Determine the net translation matrix.

1. Net Translation Vector: `T_net = T1 + T2`
2. Tx = -3 + 8 = **5**, Ty = 4 + (-2) = **2**. So, T_net(5, 2)

```
// Homogeneous Matrix M_net = M2 * M1

[ 1 0 8 ]   [ 1 0 -3 ]   [ 1 0 5 ]
[ 0 1 -2 ] × [ 0 1 4 ] = [ 0 1 2 ]
[ 0 0 1 ]   [ 0 0 1 ]   [ 0 0 1 ]

// Conclusion:
Translation is commutative. Order does NOT matter.
M1 * M2 would yield the same T_net(5, 2).
```

2D Rotation: Derivation & Matrix Form

Rotating a point P(x, y) about the origin by angle θ to P'(x', y')

Mathematical Derivation

1. Represent original point P(x, y) using polar coordinates (r, φ):

$$x = r\cos\phi, \quad y = r\sin\phi$$

2. Rotate by angle θ to get new point P'(x', y') with angle (φ + θ):

$$x' = r\cos(\phi+\theta), \quad y' = r\sin(\phi+\theta)$$

3. Expand using trigonometric addition formulas:

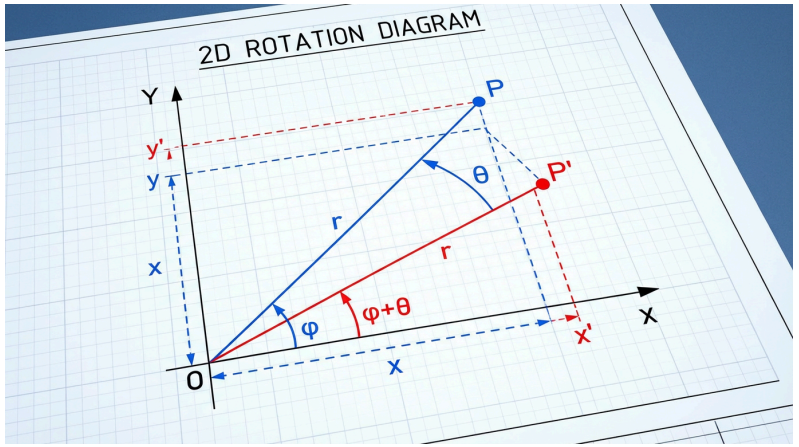
$$x' = r(\cos\phi\cos\theta - \sin\phi\sin\theta) = (r\cos\phi)\cos\theta - (r\sin\phi)\sin\theta$$
$$y' = r(\sin\phi\cos\theta + \cos\phi\sin\theta) = (r\sin\phi)\cos\theta + (r\cos\phi)\sin\theta$$

4. Substitute x and y back into equations:

$$x' = x\cos\theta - y\sin\theta \qquad y' = x\sin\theta + y\cos\theta$$

HOMOGENEOUS MATRIX REPRESENTATION

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



Key Facts

- **Positive θ:** Counter-clockwise rotation
- **Negative θ:** Clockwise rotation (sin changes sign)
- **Pivot Point:** Derivation assumes rotation about Origin (0,0)

2D Rotation: Numerical Examples

Solving standard and pivot-point rotation problems step-by-step

BASIC LEVEL

Rotation About Origin

Problem: Rotate point P(1, 0) by angle θ = 90° counter-clockwise about the origin (0,0).

1. Identify values: `x=1, y=0, θ=90°`
2. Trig values: `cos90°=0`, `sin90°=1`
3. Apply formulas:

x' = x cosθ - y sinθ	1(0) - 0(1) = **0**
y' = x sinθ + y cosθ	1(1) + 0(0) = **1**

Result: P'(0, 1)

ADVANCED LEVEL

Pivot Point Rotation

Problem: Rotate point P(3, 2) by 45° about pivot point C(1, -1).

1. Strategy: Translate C to Origin → Rotate 45° → Translate Back.
2. Trig: cos45° = sin45° ≈ 0.707

```
// Step 1: Translate P relative to C
x_t = 3 - 1 = 2, y_t = 2 - (-1) = 3



// Step 2: Rotate (2, 3)
x_r = 2(0.707) - 3(0.707) = -0.707
y_r = 2(0.707) + 3(0.707) = 3.535



// Step 3: Translate Back (+1, -1)
X' = 0.293                                  Y' = 2.535
```

2D Scaling: Theory and Matrix Form

Resizing objects by expanding or compressing dimensions relative to the origin

Theory

Scaling alters the size of an object by multiplying the coordinate values by scaling factors S_x and S_y. This operation either expands or compresses the object's dimensions.

Algebraic Equations:

Given original coordinates (x, y) and scaling factors (S_x, S_y):

$$x' = x \cdot S_x \qquad y' = y \cdot S_y$$

If $S > 1$, the object enlarges. If $0 < S < 1$, it shrinks. If $S_x = S_y$, scaling is uniform.

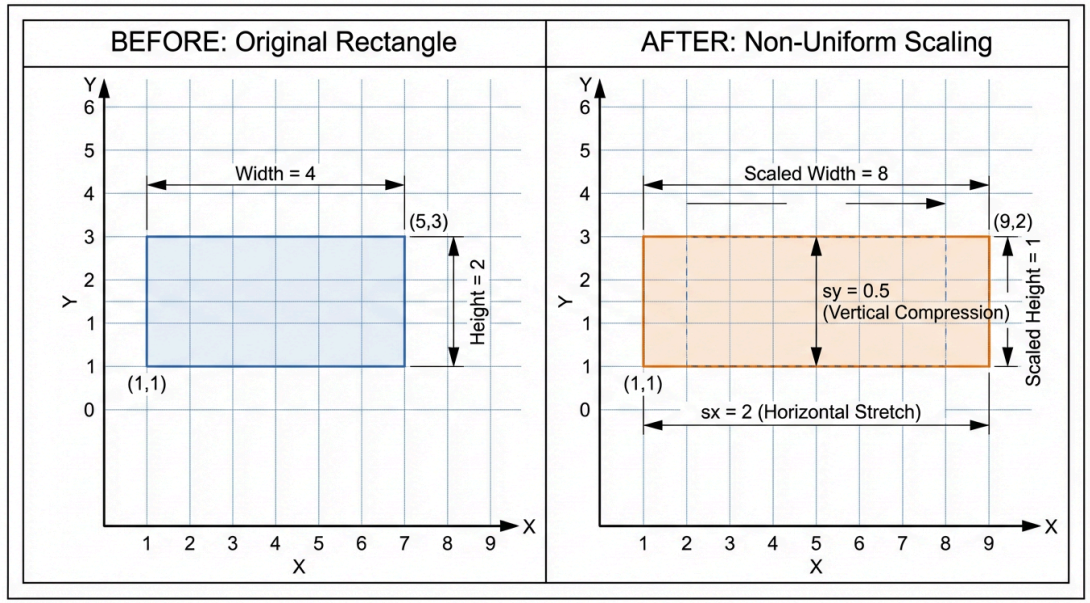
Matrix Representation

Using homogeneous coordinates, scaling can be expressed as a matrix multiplication:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Fixed Point Scaling: To scale about a fixed point (x_f, y_f):

`Translate` $T(-x_f, -y_f)$ $\rightarrow$ `Scale` $S(S_x, S_y)$ $\rightarrow$ `Translate` $T(x_f, y_f)$



Non-Uniform Scaling

2D Scaling: Numerical Examples

Solving uniform and differential scaling problems with matrices

BASIC LEVEL

Non-Uniform Rectangle Scaling

Problem: Scale a rectangle with corners A(0,0), B(2,0), C(2,1), D(0,1) using factors **Sx = 2** and **Sy = 0.5** relative to the origin.

1. Identify scaling matrix S: Diagonal [2, 0.5, 1]
2. Apply transformation: `x' = x · Sx, y' = y · Sy`

Vertex	Operation	Result
A(0, 0)	(0·2, 0·0.5)	**A'(0, 0)**
B(2, 0)	(2·2, 0·0.5)	**B'(4, 0)**
C(2, 1)	(2·2, 1·0.5)	**C'(4, 0.5)**
D(0, 1)	(0·2, 1·0.5)	**D'(0, 0.5)**

Note: The rectangle becomes wider (2x) and shorter (0.5x).

ADVANCED LEVEL

Scaling About a Fixed Point

Problem: Scale point **P(3, 4)** by factors **Sx = 1.5, Sy = 0.75** with respect to fixed point **F(1, 2)**.

1. Translate F to origin: T(-1, -2)
2. Scale at origin: S(1.5, 0.75)
3. Translate back: T(1, 2)

```
// Composite Matrix M = T(F) · S · T(-F)

⎡1  0  1⎤   ⎡1.5   0   0⎤   ⎡1  0  -1⎤
⎢0  1  2⎥ · ⎢ 0   .75  0⎥ · ⎢0  1  -2⎥
⎣0  0  1⎦   ⎣ 0    0   1⎦   ⎣0  0   1⎦

// Calculation Steps:
1. P' = P - F = (3-1, 4-2) = (2, 2)
2. P'' = P' · S = (2·1.5, 2·0.75) = (3, 1.5)
3. Result = P'' + F = (3+1, 1.5+2) = (4, 3.5)
```

2D Reflection Transformations

Theory and Matrix Forms for Axis and Origin Mirroring

Reflection Concept

A reflection creates a mirror image of an object across an axis. It preserves distance but changes chirality (handedness). It can be modeled as scaling with a negative factor.

About X-Axis (y = 0)

Maps P(x, y) → P′(x, -y)

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

About Y-Axis (x = 0)

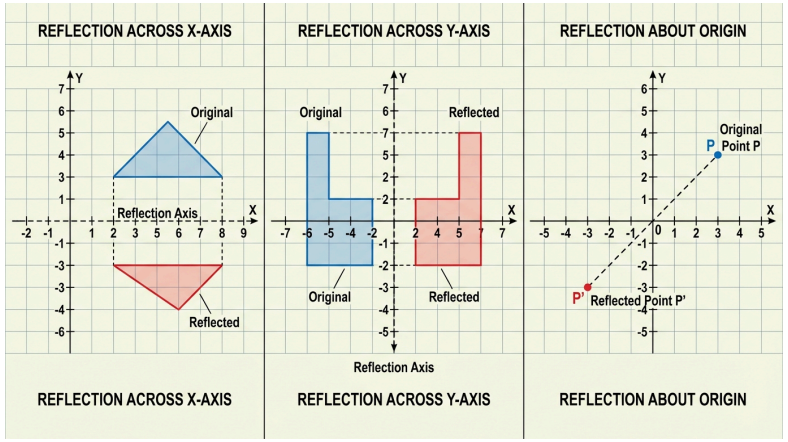
Maps P(x, y) → P′(-x, y)

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

About Origin (0, 0)

Maps P(x, y) → P′(-x, -y)

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Visualizing Reflections: Mirroring across Coordinate Axes and Origin

2D Shearing: X-Shear and Y-Shear

Distorting shape geometry by shifting coordinates along axes

X-Shear (Horizontal Shear)

X-Shear preserves the Y coordinate but shifts the X coordinate proportional to the value of Y.

$$x' = x + sh_x \cdot y \qquad y' = y$$

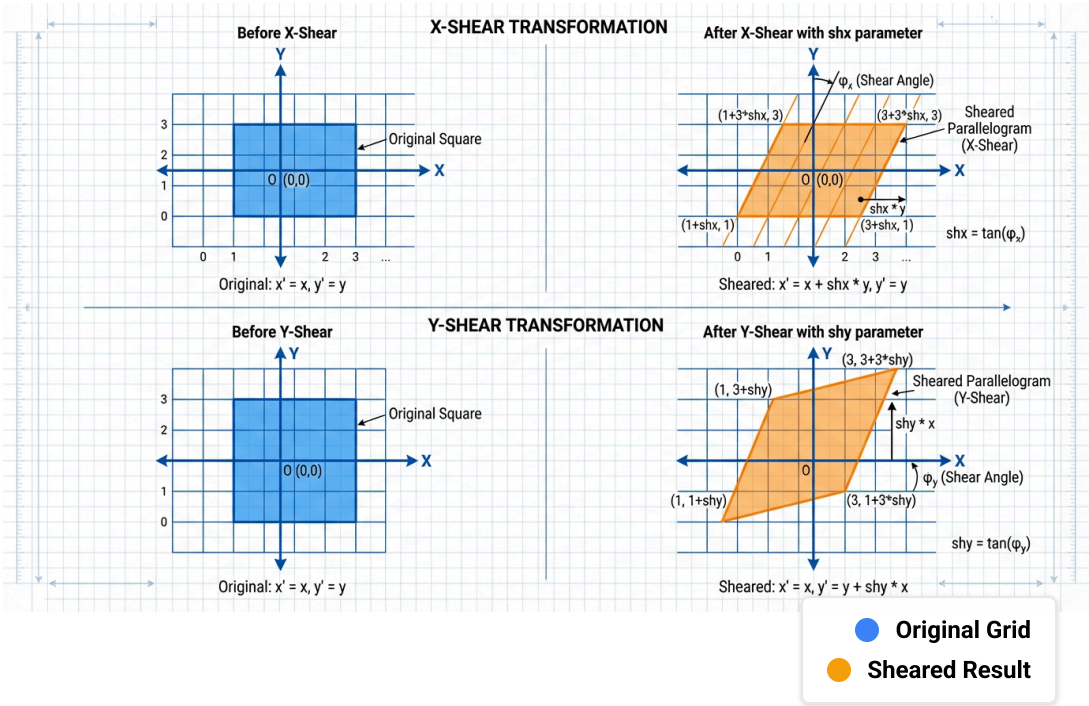
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Y-Shear (Vertical Shear)

Y-Shear preserves the X coordinate but shifts the Y coordinate proportional to the value of X.

$$x' = x \qquad y' = y + sh_y \cdot x$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



Composite 2D Transformations

Combining multiple operations into a single transformation matrix

Order Matters

Matrix multiplication is **not commutative**: `A · B ≠ B · A`. Transformations are applied right-to-left: `P' = (T · R · S) · P` means Scale first, then Rotate, then Translate.

Concatenation

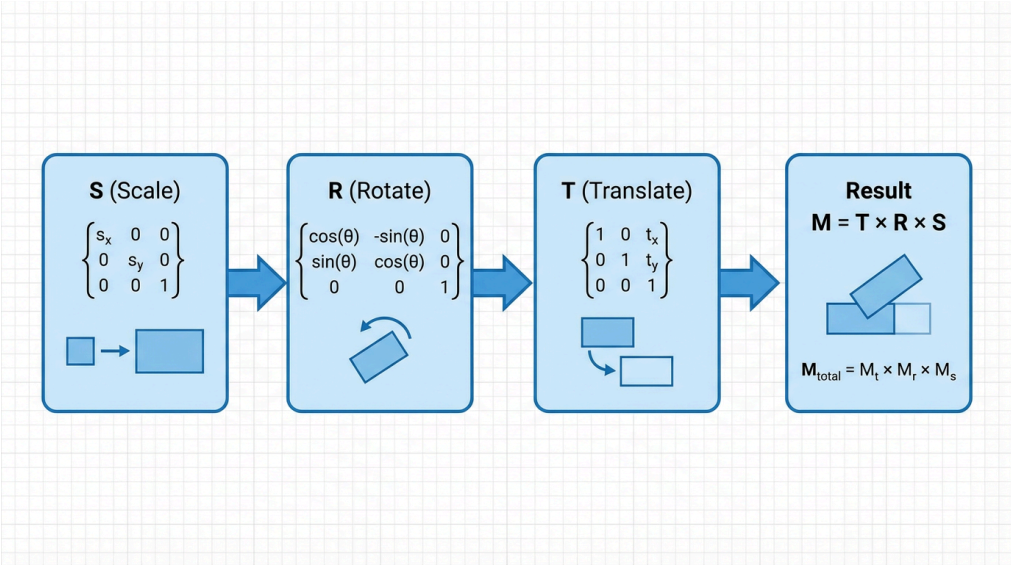
Multiple matrices `M1...Mn` can be multiplied to form a composite matrix `M = Mn ·...· M1`. This reduces computation for complex scenes.

Pipeline Efficiency

Instead of transforming every point `n` times, we compute `M` once and apply it to all points. Essential for real-time graphics and animation.

Transformation Pipeline Visualization

`M = T(tx,ty) · R(θ) · S(sx,sy)`



Common Composite Operation:

Rotation about Pivot Point (xp, yp):

`M = T(xp, yp) · R(θ) · T(-xp, -yp)`

Composite 2D Transformations: Worked Examples

Solving multi-step transformation problems using matrix concatenation

BASIC LEVEL

Scale then Translate

Problem: Scale a square with vertex C(2,2) by 2x uniformly, then translate by T(3, 1). Find new C'.

1. Form Matrices: S(sx=2, sy=2) and T(tx=3, ty=1).
2. Compute Composite M = T × S (Order matters! Apply S first).
3. Transform point C(2,2) using M.

```
// Composite Matrix M = T * S

[1 0 3]   [2 0 0]   [2 0 3]
[0 1 1] × [0 2 0] = [0 2 1]
[0 0 1]   [0 0 1]   [0 0 1]

// Calculate Point C'
x' = 2(2) + 3 = 7
y' = 2(2) + 1 = 5
Result: C'(7, 5)
```

ADVANCED LEVEL

Reflect, Rotate & Translate

Problem: Transform P(4, -2) by: 1) Ref(y=x), 2) Rot(30° CCW), 3) Trans(2, -1).

1. Order: M = T(2,-1) × R(30°) × Ref(y=x)
2. Ref(y=x): Swaps (x,y) → (y,x). P'(-2, 4)
3. Rot(30°): cos30≈0.87, sin30=0.5. Apply to P'

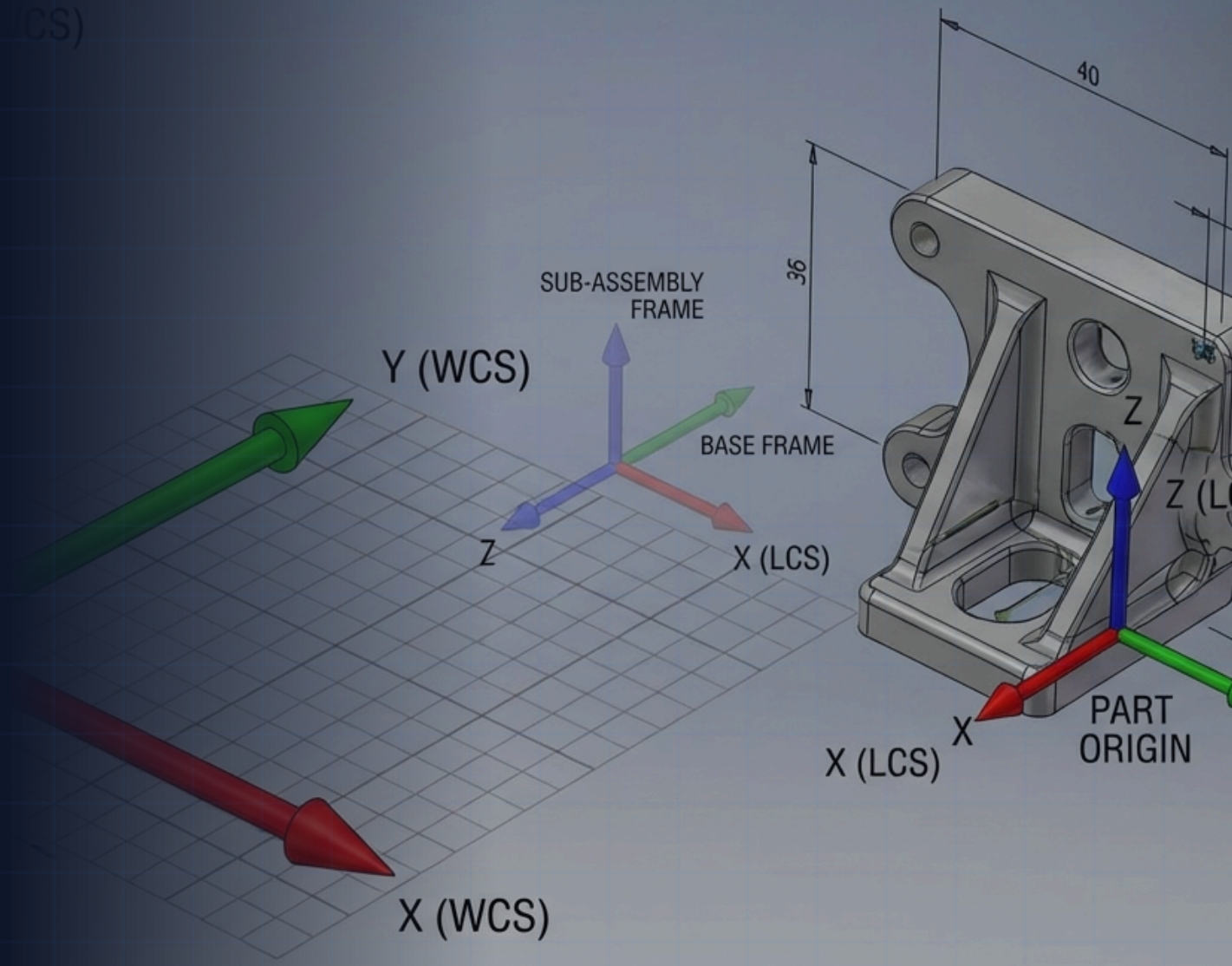
```
// Step-by-Step Calculation:
1. Ref P(4,-2) → P'(-2, 4)
2. Rot P'(-2, 4):
  x'' = -2(0.87) - 4(0.5) = -1.74 - 2 = -3.74
  y'' = -2(0.5) + 4(0.87) = -1 + 3.48 = 2.48
3. Trans P''(-3.74, 2.48) by (2, -1):
  x''' = -3.74 + 2 = -1.74
  y''' = 2.48 - 1 = 1.48
Final Result P'''(-1.74, 1.48)
```

SECTION 03

3D TRANS FORMATIONS

> Extending coordinate geometry into spatial dimensions for complex CAD modeling.

- 3D Coordinate Systems (X, Y, Z)
- 4x4 Homogeneous Matrices
- Rotation about Major Axes



3D Translation: Matrix Form and Examples

Displacement of objects in three-dimensional space

A 3D translation shifts a point `P(x, y, z)` to `P'(x', y', z')` by adding translation distances t_x, t_y, t_z.

ALGEBRAIC EQUATIONS

$$x' = x + t_x \qquad y' = y + t_y \qquad z' = z + t_z$$

HOMOGENEOUS MATRIX (4X4)

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

NUMERICAL EXAMPLE

Given:

$P(-1, 2, 3)$

$T = [4, -1, 2]$

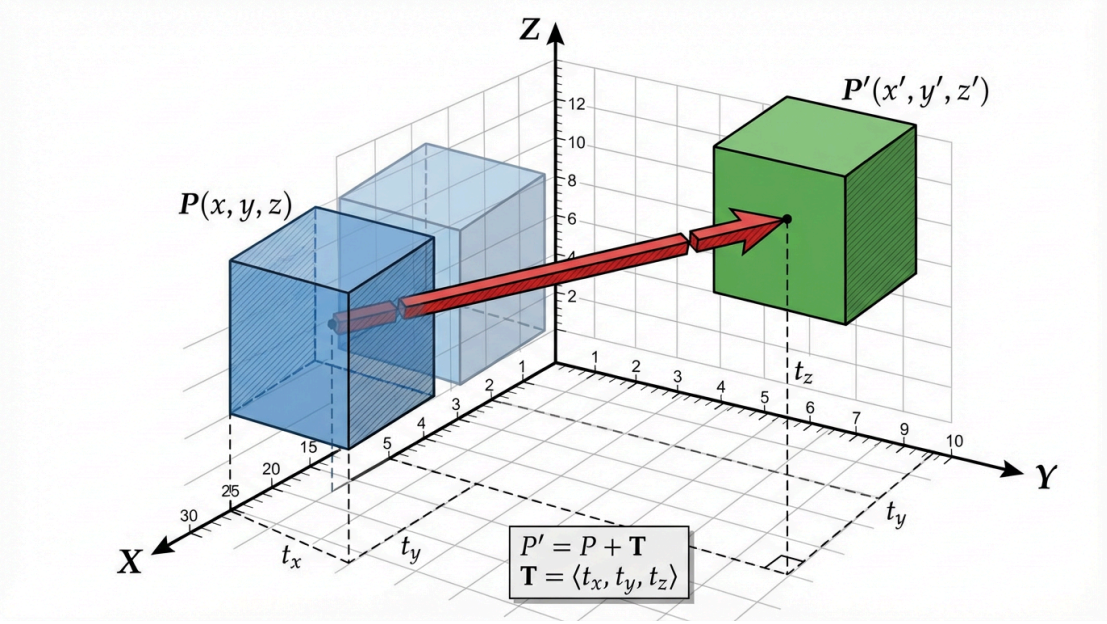
Calculation:

$x' = -1 + 4 = 3$

$y' = 2 - 1 = 1$

$z' = 3 + 2 = 5$

Result: $P'(3, 1, 5)$



3D Displacement

3D Rotation About X-Axis

Derivation and Homogeneous Matrix Representation

1. Invariant Coordinate

In X-axis rotation, the X-coordinate remains unchanged as the rotation occurs perpendicular to it.

$$x' = x$$

2. Planar Rotation (Y-Z Plane)

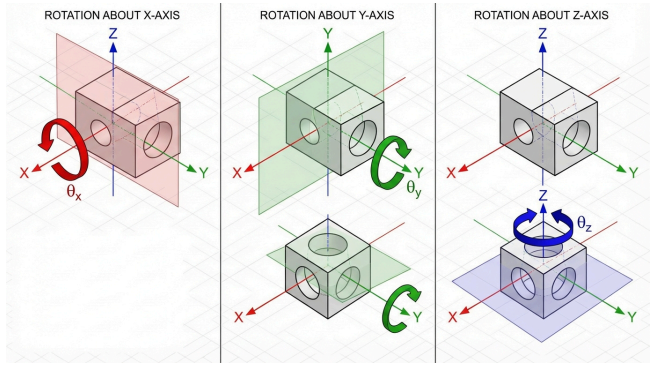
The rotation occurs in the Y-Z plane. By convention (Right-Hand Rule), Y acts as the horizontal axis and Z as the vertical axis.

$$y' = y\cos\theta - z\sin\theta$$

$$z' = y\sin\theta + z\cos\theta$$

3. Homogeneous Matrix (R_x)

$$[R_x] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



NUMERICAL EXAMPLE

Problem: Rotate point $P(1, 2, 3)$ by $\theta = 90^\circ$ about the X-axis.

$\cos 90^\circ = 0, \quad \sin 90^\circ = 1$

$x' = 1$

$y' = 2(0) - 3(1) = -3$

$z' = 2(1) + 3(0) = 2$

Result: P'(1, -3, 2)

3D Rotation About Y-Axis

Rotating a point P(x, y, z) in the X-Z plane while Y remains constant

1. Invariant Coordinate

Rotation about the Y-axis does not change the Y-coordinate.

$$y' = y$$

2. Planar Rotation (X-Z Plane)

The Z-axis rotates toward the X-axis (Right-Hand Rule).

$$z' = z\cos\theta - x\sin\theta$$

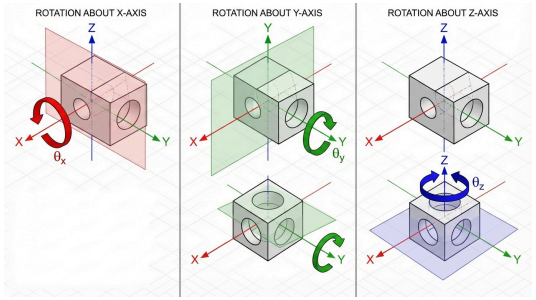
$$x' = z\sin\theta + x\cos\theta$$

3. Matrix Form

Align variables x, y, z to form the homogeneous matrix.

ROTATION MATRIX (R_Y)

$$[R_y] = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



NUMERICAL EXAMPLE

Problem: Rotate $P(1, 2, 3)$ by $\theta = 90°$ about Y-axis.

$\cos 90° = 0, \quad \sin 90° = 1$

```
Calculation:
x' = 1(0) + 3(1) = 3
y' = 2 (unchanged)
z' = -1(1) + 3(0) = -1
Result P' = (3, 2, -1)
```

3D Rotation About Z-Axis

Derivation, Matrix Form, and Numerical Example

Mathematical Derivation

1. **Concept:** Rotating a point $P(x, y, z)$ about the Z-axis implies that the z-coordinate remains unchanged ($z' = z$). The rotation occurs purely in the XY plane.

2. **2D Analogy:** The equations are identical to 2D rotation for x and y:

$$x' = x\cos\theta - y\sin\theta$$

$$y' = x\sin\theta + y\cos\theta$$

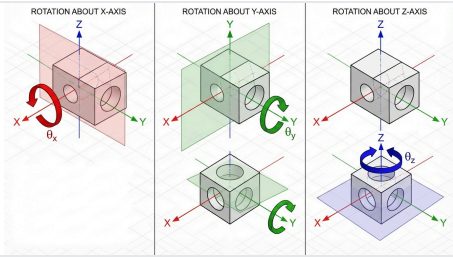
$$z' = z$$

3. **Homogeneous Coordinates:** Extend to 4x4 matrix with $w = 1$.

Z-AXIS ROTATION MATRIX (R_Z)

$$[R_z] = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

VISUALIZATION



Rotation occurs in XY plane (Right-most diagram)

NUMERICAL EXAMPLE

Problem: Rotate $P(1, 2, 3)$ by $\theta = 90°$ about Z-axis.

$\cos 90° = 0, \sin 90° = 1$

$x' = 1(0) - 2(1) = -2$
$y' = 1(1) + 2(0) = 1$
$z' = 3$

Result: $P'(-2, 1, 3)$

3D Scaling and Shearing

Geometric deformations in three-dimensional space

3D Scaling

Resizes object along axes using scale factors s_x, s_y, s_z. A diagonal matrix operation.

$$S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$x' = x \cdot s_x, \quad y' = y \cdot s_y, \quad z' = z \cdot s_z$$

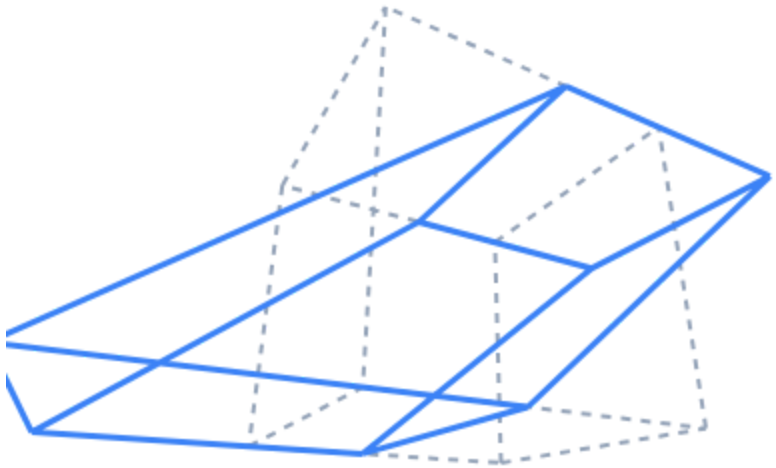
3D Shearing (X-Shear)

Distorts shape by shifting the X-coordinate based on Y and Z values.

$$Sh_x = \begin{bmatrix} 1 & sh_{xy} & sh_{xz} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$x' = x + y \cdot sh_{xy} + z \cdot sh_{xz}$$

Visualization: X-Shear



Original Sheared

Composite 3D Transformations

Sequential application of translation, rotation, and scaling matrices

Order is Critical

Matrix multiplication is non-commutative ($A \cdot B \neq B \cdot A$). Rotating then translating yields a different result than translating then rotating.

Arbitrary Pivot

To transform about point P:

1. Translate P to Origin $T(-P)$
2. Apply Transform R/S
3. Translate back $T(P)$

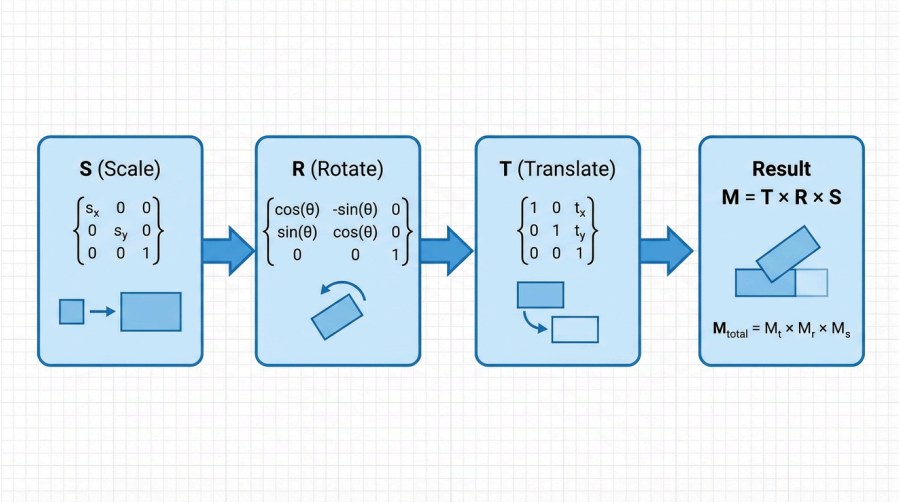
Concatenation

For column vectors apply right-to-left:

$P' = M_n \cdot \ldots \cdot M_1 \cdot P$

Compute unified matrix **M** once.

Transformation Pipeline Visualization



→ M = T · R · S

Viewing Pipeline & Window-to-Viewport

Mapping 3D World Coordinates to 2D Screen Coordinates

The Viewing Pipeline

World Coordinates (WCS)
Scene construction

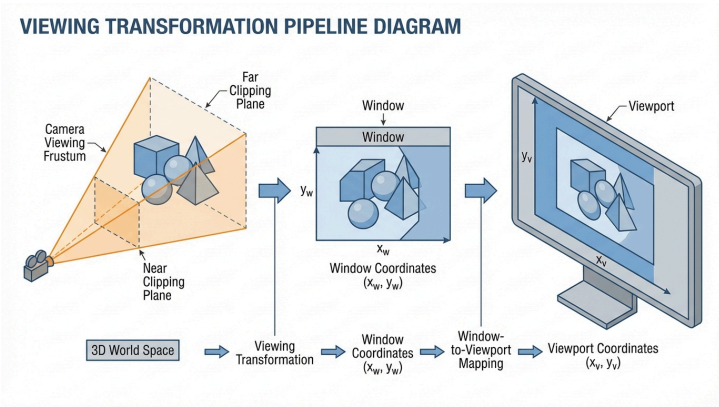
Viewing Coordinates (VCS)
Camera reference frame

Clipping
Remove invisible geometry

Normalized Device (NDC)
Map to [0,1] or [-1,1] range

Screen Coordinates (SCS)
Rasterization to pixels

Window-to-Viewport Transformation



MAPPING FORMULAS

Objective: Map point (xw, yw) in Window to (xv, yv) in Viewport.

Scaling Factors:

$$s_x = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}}$$

$$s_y = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}}$$

Transformation Equations:

$$xv = xv_{min} + (xw - xw_{min}) \cdot s_x$$

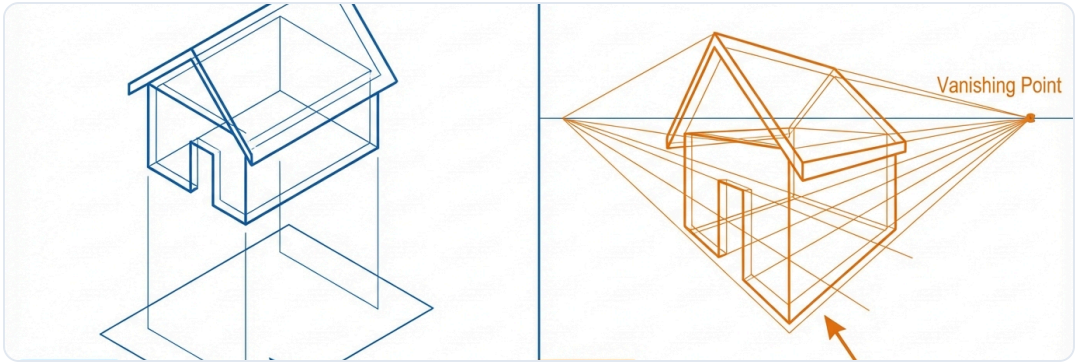
$$yv = yv_{min} + (yw - yw_{min}) \cdot s_y$$

** Maintains relative position ratios.*

Projection Transformations

Converting 3D Models to 2D Views: Parallel vs. Perspective

Parallel Projection



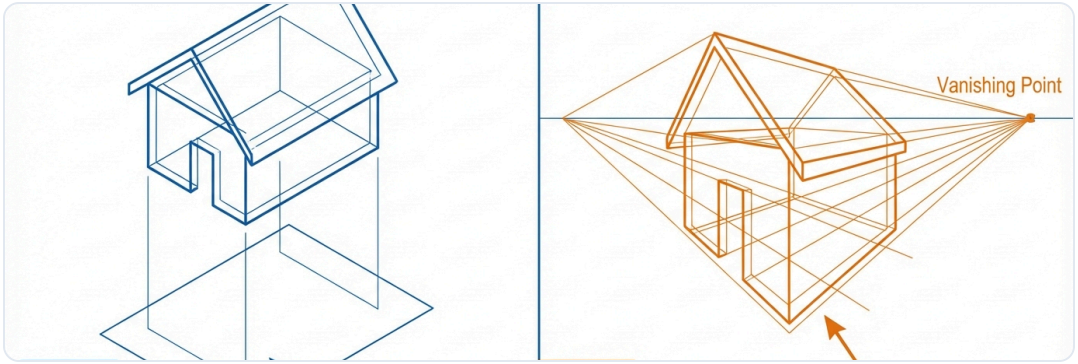
Preserves Parallelism: Parallel lines remain parallel.

ORTHOGRAPHIC MATRIX (SIMPLE)

$$M_{ortho} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

*Drops Z-coordinate for viewing on XY plane

Perspective Projection



Realistic Depth: Objects appear smaller as they get further away (Foreshortening).

PERSPECTIVE MATRIX (SIMPLE)

$$M_{persp} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}$$

*d = distance to projection plane

Applications in CAD Systems

Practical implementation of geometric transformations in engineering

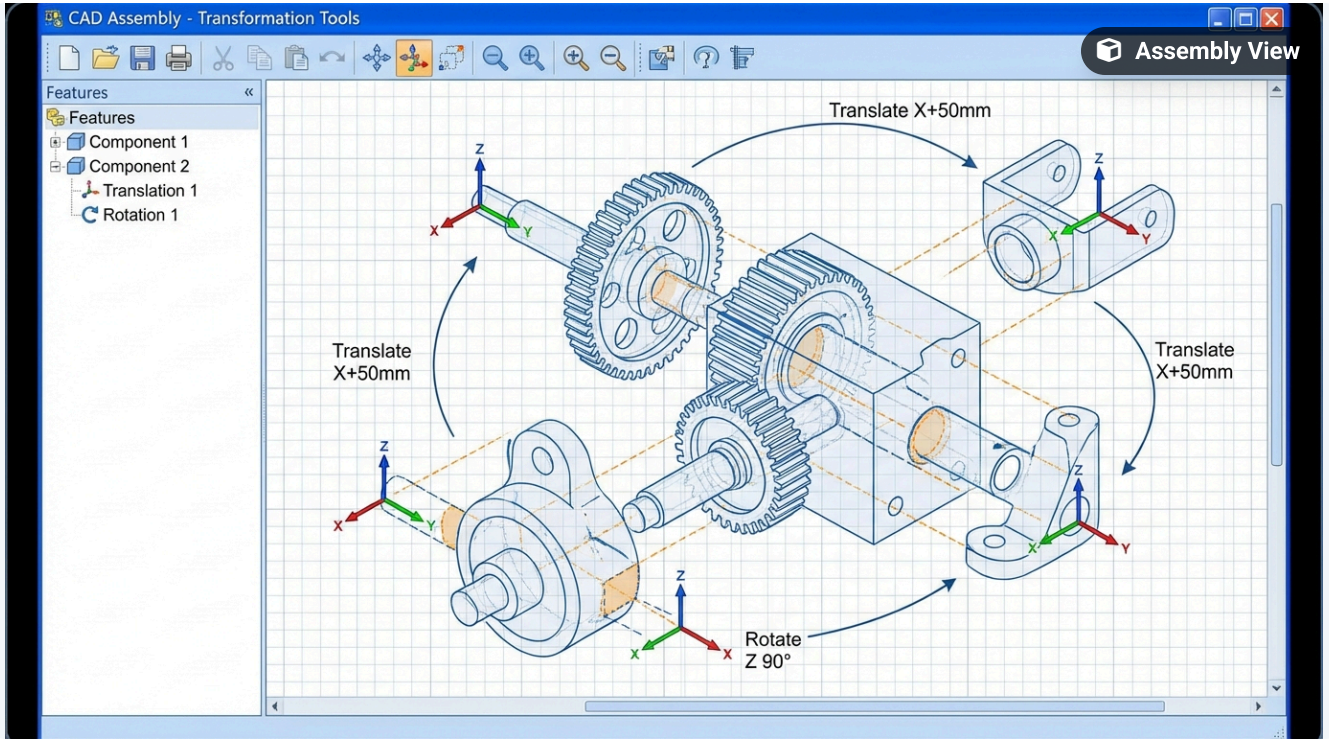


Figure: Hierarchical assembly modeling where each component's local position (MCS) is transformed into the global assembly space (WCS).

Assembly Modeling

Placement of parts involves calculating translation (T) and rotation (R) matrices to satisfy mating constraints like coincidence and concentricity.

Robotics & Kinematics

Forward and inverse kinematics use chains of homogeneous transformation matrices to determine end-effector position and orientation precisely.

Views & Animation

Exploded views translate parts along specific axes. Walkthroughs update the Viewing Coordinate System (VCS) dynamically for user interaction.

CAM & Tool Paths

Transforming tool coordinates to workpiece coordinates for 5-axis machining involves complex composite 3D transformations for accurate cutting.

Important Facts & Summary

Key takeaways for implementing CAD transformations effectively

Order Criticality

Matrix multiplication is non-commutative. Applying operations in the wrong order (e.g., Rotation then Translation vs Translation then Rotation) yields completely different results.

> Remember: `M = T · R · S` applies S, then R, then T.

Homogeneous Power

Using (n+1) dimensions allows translation to be represented as matrix multiplication, unifying it with scaling and rotation. This enables efficient composite matrices.

> 3D Point: `(x, y, z, 1)`

Units & Conventions

Always normalize vectors before applying transformations. Be mindful of trigonometric functions expecting **Radians** versus Degrees in most programming libraries.

> deg = rad × (180/π)

Aspect Ratio Preservation

In Window-to-Viewport transformations, ignoring the aspect ratio of the target display device results in distorted (stretched or squashed) geometry. Ensure `AR_window = AR_viewport` or use uniform scaling.

Numerical Stability

Deeply nested composite transformations (e.g., in robotic arms with many joints) can accumulate floating-point errors. Use double precision and re-normalize rotation matrices occasionally to maintain orthogonality.

Practice Problems

Test your understanding of CAD transformations at varying difficulty levels

Basic Operations

LEVEL 1

Problem 1: Translation

Translate point `P(-2, 5)` by vector `T(7, -3)`.

Goal: Find the new coordinates P'.

Problem 2: Scaling

Scale triangle defined by origin, (2,0), (0,3) by factors `sx=0.5, sy=2` about the origin.

Goal: Determine the new vertex coordinates.

Pivot & Composition

LEVEL 2

Problem 3: Pivot Rotation

Rotate point `P(4, -1)` by `-30°` about the pivot point `C(2, 2)`.

Hint: Use T(C) · R(-30°) · T(-C).

Problem 4: Composite Matrix

Build composite matrix `M` for: Scale(2,1) → Rotate(90°) → Translate(1,-2).

Goal: Find P' if initial P is (1,1).

3D & Pipelines

LEVEL 3

Problem 5: 3D Sequence

Transform `P(1, 2, 3)` by rotating 45° about Z-axis, then 30° about X-axis, then translate by (-2, 1, 4).

Goal: Set up the 4x4 matrix multiplication chain.

Problem 6: Window-to-Viewport

Map window `[0,100]x[-50,50]` to viewport `[200,100]` top-left to `[1000,700]` bottom-right.

Goal: Derive mapping equations for point (xw, yw).